HERC 2000-03

Development of a Basic Flight Instruction Tutoring System (BFITS) Research Station

Dr. James C. Miller
Michael T. Kirk
Human-Environmental Research Center

C1C John S. Flynn, C1C Morgan P. Hurt
C1C Jeffrey C. Schlueter, C1C Matthew W. Stewart
C1C Martin W. Weeks III
Department of Behavioral Sciences and Leadership

United States Air Force Academy
Colorado Springs, Colorado 80840

April 2000

DEAN OF THE FACULTY
UNITED STATES AIR FORCE ACADEMY
COLORADO 80840

HERC 2000-03

This report entitled *Development of a Basic Flight Instruction Tutoring System (BFITS) Research Station* is presented as a competent treatment of the subject, worthy of publication. The United States Air Force Academy vouches for the quality of the research, without necessarily endorsing the opinions and conclusions of the authors. Therefore, the views expressed in this article are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the US Government.

This report has been cleared for open publication and public release by the appropriate Office of Information in accordance with AFI 61-202 and USAFA FOI 190-1. This report may have unlimited distribution.

Alice J. Chen

ALICE J. CHEN, Lt Col, USAF
Director of Faculty Research

31 May 2000
Date

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
24 April 2000		January 1999 - April 2000

4. TITLE AND SUBTITLE
Development of a Basic Flight Instruction Tutoring System (BFITS) Research Station

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
James C. Miller, Ph.D., CPE, Michael T. Kirk, John S. Flynn, Morgan P. Hurt, Jeffrey C. Schlueter, Matthew W. Stewart, Martin W. Weeks III

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Human-Environmental Research Center,
and Department of Behavioral Sciences and Leadership
Dean of the Faculty
USAF Academy, Colorado 80840

8. PERFORMING ORGANIZATION REPORT NUMBER
HERC 2000-03

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution A: Approved for public release. Distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The Basic Flight Instruction Tutoring System (BFITS) was designed to observe and track the behavior of students as they attempt to learn basic flight procedures (Benton et al., 1992). Several modifications were required to adapt the BFITS to the needs of researchers at the United States Air Force Academy. These modifications were accomplished by a team of five cadets enrolled in Behavioral Sciences 473, Human Factors Engineering, and their instructor. A limited test and evaluation assured that the BFITS research station worked as desired. The BFITS is a rich resource for investigations of fundamental learning processes associated with novice pilot learning. It may be used extensively as a tool to support faculty and cadet investigations of those processes.

15. SUBJECT TERMS
Learning, basic flying skills, pilot, simulation, research, military academy, USAFA, HERC

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. James C. Miller
Unclassified	Unclassified	Unclassified	Unl	40	19b. TELEPHONE NUMBER *(Include area code)* (719) 333-2804

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Academy produced large numbers of UPT entrants each year. HERC Project 9901AIR provided a mechanism to study in a systematic manner the principles of airmanship and what individual characteristics constituted a "good" pilot, including the study of the acquisition of basic flying skills. This report describes the establishment within 9901AIR of a flight simulation tool to study this kind of skill acquisition.

The Basic Flight Instruction Tutoring System (BFITS) observed and tracked the behavior of students as they attempt to learn basic flight procedures (Benton et al., 1992). It taught the basic concepts and principles of flying, and then allowed students to apply what they had learned in the simulator setting of the program. Development work for and field validation of the BFITS were performed by contractors supported by the Air Force Armstrong Laboratory (now, the Human Effectiveness Directorate of the Air Force Research Laboratory; AFRL/HE). The BFITS was, apparently, not in use at any other Air Force laboratory. Its software provided data that supported fundamental investigations of learning.

Several modifications were required to adapt the BFITS to the needs of researchers at the United States Air Force Academy. These modifications were accomplished by a team of five cadets enrolled in Behavioral Sciences 473, Human Factors Engineering, and their instructor, following a process that included literature review, system analyses, system design, and test and evaluation. A 90-megahertz Pentium personal computer host was set up and complemented with simulated aircraft yoke and rudder pedals. The cadets created easily understood instructions for users. The instructor wrote software that made BFITS academic lesson and flying performance data viewable in commercial spreadsheets.

A limited test and evaluation assured that the BFITS research station worked as desired. The project used two independent groups who flew the first 14 of the 31 BFITS lessons: group one (novice), no prior fixed wing flying or flight simulator experience; and group two (experienced), who had at least completed their first solo in the soaring program. The hypothesis was simply that experienced users would demonstrate better performance in lessons and in simulated flight than novices.

The BFITS is a rich resource for investigations of fundamental learning processes associated with novice pilot learning. It may be used extensively as a tool to support faculty and cadet investigations of those processes. The BFITS software allows many manipulations:

- The field of view of the pilot may be altered for simulated flight; this will allow an attempt to investigate a theory of visual field requirements in simulation put forth by Stan Roscoe several decades ago.
- Questions and answers may be rephrased and entire lessons re-written.
- The instrument panel may be re-organized.
- The pass-fail criteria for flight performance may be altered.

INTRODUCTION

The Academy produced large numbers of UPT entrants each year. HERC Project 9901AIR provided a mechanism to study in a systematic manner the principles of airmanship and what individual characteristics constituted a "good" pilot, including the study of the acquisition of basic flying skills. This report describes the establishment within 9901AIR of a flight simulation tool to study this kind of skill acquisition.

This report documents the creation of a workstation for learning research by a team of cadets enrolled in Dr. Miller's section of Behavioral Sciences 473, Human Factors Engineering, during the spring semester of 1999 at the US Air Force Academy. The research workstation was to be used in subsequent investigations of the learning of basic flying skills. The cadets were to follow a highly structured process that included literature review, system analyses, system design, and test and evaluation. The assignment was to:

Create a flight training research system

Background: The Human-Environmental Research center (HERC) has acquired software entitled the Basic Flight Instruction Tutoring System (BFITS). The software was created by Technology Systems Inc. under AF contract. BFITS "teaches a subset of the basic concepts and principles of flying and then provides the opportunity to practice the application of these in the flight simulator component of the tutor." Flying performance data are stored in ASCII text files. Reportedly, a validation study by Koonce and others suggested that the use of BFITS can reduce the number of flight hours to first solo by up to one-third. The HERC requires that this software become the central part of a research system that supports investigations of the way new pilot trainees learn to fly.

System Requirements:
The system shall

- Be configured to support learning and training research by USAFA cadets and faculty
- Include applicable, easily understood instructions for investigators for the use of the system as a research tool
- Allow pilot performance data to be viewed in spreadsheet form
- Be supported by all necessary hardware
- Be located in a suitable environment for the conduct of research
- System function shall be demonstrated by conducting one or more demonstration research projects, including hypothesis testing, using one or more subject groups, each of which contains at least six subjects with no prior pilot experience and who are not members of the design team, and an investigator who is not part of the design team. Human use approval will be needed.

Literature Review

Schneider (1990) focused on the fallacies of training high performance skills, such as flying an airplane. The first fallacy was the idea that 'practice makes perfect'. Schneider stated that, for tasks such as memorizing a telephone number or learning the way to work, practice makes perfect. However, when dealing with those who work in the air traffic control tower or in the cockpit of an airplane, "this assumption does not prove to be a valid generalization for high-performance training" (p. 298). In some cases, Schneider claimed, practice may not help at all. Practice makes perfect when the same routine and steps are taken in the same order repeatedly. However, when flying an airplane, the exact sequence of events rarely repeats. In addition, when operating an air traffic control tower, the situation changes continually. Thus, for tasks such as flying an airplane or operating an air traffic control tower, achieving perfection by practice is nearly impossible because different situations call for different decisions.

Another fallacy was that "it is best to train a skill in a form similar to the final execution of the skill" (p. 299). The problem with this belief was that trainees would never acquire the instruction in the steps leading up to the final skill execution. The steps leading up to the final task may not always be the same. No one can predict what is going to happen next. In this case, if someone has not been properly trained in the steps leading up to the final stage, they will be inadequately prepared for the event or emergency.

The third fallacy was that "skill learning is intrinsically motivated" (p. 300). This fallacy assumed that the person did not require any extrinsic motivating factors. One example, provided by Schneider, dealt with air traffic controllers. An air traffic controller's career was based upon his performance in the tower and his ability to apply the skill that he learned. If he did not performing the job correctly, he would lose the job and any monetary rewards that came with it. This was a strong, extrinsic motivating factor. If people are not rewarded for their jobs, why would anyone want to acquire that skill in the first place?

The next fallacy was that "people should train for accurate performance" (pp. 300-301). Specifically, trainers would train air traffic controllers one function at a time, making sure that their performance on this task was perfect. This is not how it should be. Schneider stated "In many skill-training programs, the goal should be to obtain acceptable accuracy on a component skill while allowing attention to be allocated to other components of the task" (p. 301). As an air traffic controller or pilot, it is imperative that the person is capable of allocating his attention to different tasks at the same time. Without resource allocation like this, the person would never be able to take upon the responsibility of a job requiring high-performance skill.

The fifth fallacy was that "initial performance is a good predictor of trainee and training program success" (p. 301). For obvious reasons, most people are not going to perform well during their first couple attempts at a high-performance skill. The trainee must first be exposed to the skill for a while (preferably 100-200 hours) and learn all about the system before an evaluation can be made.

The final fallacy was that "once the learner has a conceptual understanding of the system, proficiency will develop in the operational setting" (p. 301). Specifically, if someone did

well in the classroom during Undergraduate Pilot Training (UPT), this fallacy suggested that he or she would also do well when it was time to apply those concepts in the cockpit. But as we all know, sitting in a classroom and rehashing knowledge in an air conditioned room is completely different than doing it in a multi-million dollar aircraft with your life on the line. The classroom can only prepare you to a certain extent.

Stark (1989) focused on simulator design. Simulators viewed as being used for three primary purposes: "to facilitate in experimenting with a system without having to actually build it; support and apply (past) research dealing with human limitations and capabilities; and simulators are used to train personnel in the operation of a system" (p. 109). Each simulator varies in effectiveness but there seems to be one quite common requirement: the simulator training must be transferable from the training environment into the operational environment. In addition to this, there are some other requirements in the simulator design. One requirement is that it must provide the most realistic environment, compared to the one in which the pilot will fly. Also, the simulator's mechanisms must be clear and easy to understand so the pilot knows what instruments stand for what. Standardization is also important. The same feedback must be given repeatedly to ensure accuracy and proper acquisition of skill. Finally, instructional interaction is pivotal. The instructor must be proficient and up to date on the simulator's operational methods. Without this capability, the training may produce negative effects.

Koonce (1998) described a validation study of the BFITS conducted by the University of Central Florida. The BFITS was a flight simulation system that runs in DOS. The system was set up with rudder pedals, a joystick or yoke, and a VGA monitor. Koonce's experiment was designed to test the effectiveness of the BFITS system on novices, those who had no previous flight experience. To test BFITS' effectiveness, Koonce analyzed the amount of "flight time to first solo, number of landings prior to first solo, and total flight time to the private pilot certificate".

There were three groups of subjects. The first was a control group that received no time in the BFITS simulator but proceeded with flight instruction until all requirements were satisfied. The second group contained 28 subjects from Taiwan, whose first language was not English. The third group held 129 English-speaking subjects.

The results showed that, compared to the control group, the English speaking subjects "had an average of 3.21 hours less dual instruction prior to their first solo flight, 14.2 fewer landings prior to first solo, and 8.46 hours less total flight time to the private pilot certificate." The Taiwanese subjects needed 0.54 more hours of dual instruction, compared to the English subjects. However, the Taiwanese had 5.26 fewer landings before first solo.

Koonce's experiment showed the effectiveness and need for a BFITS system in an introductory flight training course. Most of the results reported what one would expect, except for the Asian subjects performing fewer landings before their first solo. Koonce's explanation for this is that they had developed the perceptual-motor skills in the BFITS.

Taylor et al. (1993), at the University of Illinois Institute of Aviation, examined transfer of flight skills training with flight training subjects. This study focused on determining the optimum design and features for flight training simulators. Some of the variables tested in this study were environmental effects, dynamic responses, visual scenes, and instrumental augmentations. The primary task in the first part of this experiment was approach and landing in a light aircraft. This task was chosen due to its high level of difficulty. Since landing is one of the more dangerous parts of flying, tests done which improve landing abilities could be highly beneficial.

Taylor's experiment was conducted on 40 male and two female flight subjects from 18-30 years old with no prior flight experience. The group of subjects was broken down into 21 pairs. From each pair, one subject received flight simulator training before instruction while the other subject only received flight instruction. The subject receiving no simulator training was the control. The training device used in this experiment was the ILLIMAC flight trainer. This flight trainer is a fixed base, digital simulator equipped with full light aircraft controls and displays. The task placed upon the experimental group of subjects was to make simulated approaches on a 4-degree glideslope starting at 10,100 feet from the runway, at 635 ft altitude, and lined up with the runway centerline. The simulator provided no crosswind during the simulation. The subjects conducted the flight approach, flare, touchdown, and rollout.

The flight training itself consisted of a beginning flight course which lasted through a 16-week semester and included ground school, 6 hours of instruction in a Link GAT-1 ground based flight trainer, and 24 hours of aircraft flying, including 3 hours of solo flight. The solo flight could not be accomplished before 17 hours of flight time. The experimental subjects received two additional training sessions, which included 26 landing trials each, in the ILLIMAC flight trainer. These additional sessions were conducted immediately before intensive landing sessions with an instructor.

The results of this experiment showed that the experimental group of subjects averaged 66.0 landings prior to solo at an average of 17.6 hours of flight time. The control subjects averaged 75.8 landings before soloing and flew an average of 18.6 hours. The 9.8 difference in the number of landings was statistically significant, $t(16) = 2.21$; $p = 0.021$, one-tailed. These results showed that 2 hours of flight simulator training could reduce the number of pre-solo landings in light aircraft. Even with the limiting factor of needing 17 hours to solo, the experimental group still required significantly fewer hours to solo. Therefore, from this experiment, we can conclude that flight simulator training can aid training performance in light aircraft.

In another experiment, conducted by Park and Lee (1992), computer aided tests were used to predict flight performance of trainees. This experiment focused around determining what skills were good predictors of flight training performance. Based upon previous literature and the experience of the authors, the categories used to classify pilot functions were tracking, reaction, memory, estimation, and visual scanning. These five categories were selected because they were viewed as essential functions for a pilot to perform.

This study was conducted upon 64 senior cadets at the Korean Air Force Academy. They ranged from 22 to 24 years old. None of the subjects had prior flight experience. The apparatus used in this experiment was a computer that generated all the stimulus signals and then recorded the data. The subjects used keyboard, joystick, foot pedals, and a secondary keyboard to input their responses. The test was composed of 16 single tasks and 10 dual tasks. The single tasks were composed of tasks such as: hand and foot tracking, color and sound reaction, item recognition, memory tasks, time, speed, and size estimation, shape comparison, and position reading. The dual tasks consisted of one-hand tracking while performing the 10 tasks of reaction, memory, and estimation categories and were designed to measure time-sharing abilities.

Upon completion of the experiments, the subjects began a basic flying course that included 14 hours of dual flight time in a Cessna T-41 and a 1-hour check ride with a flight evaluator. All cadets passed this flight training and entered an intermediate flight training school. The intermediate school consisted of advanced flight training in a Cessna T-37. The training consisted of 60 hours flight time. After completing the first 25 hours of flight training, each subject was screened for the next stage of training. From the intermediate flight training, 43 subjects passed, and 20 failed.

The results of the experiment showed that the passing group generally performed better than the failing group in the single-task measures. However, there were only statistically significant differences ($p < 0.05$) between the two groups in three single task measures of foot-hand tracking, position reading, and shape comparison, and in one dual-task measure; one-hand tracking plus kinesthetic memory. From this study it was concluded that of the five factors originally believed to be good predictors of flight performance, only three factors ended up being good predictors: tracking, reaction, and memory.

This experiment was important because it provided us with direction as to what to focus flight simulator training on. Since reaction, memory, and tracking seem to be important predictors in pilot performance, it would be good to design a simulator that develops these three factors. In addition, when evaluating how effective flight simulator training is, it would be good to test how the training affected the three factors of reaction, memory, and tracking.

SYSTEM ANALYSES

Predecessor Systems Analysis

With computers advancing as quickly as they were, flight simulators were becoming more available to consumers and were becoming increasingly realistic. There were several different flight simulators available for personal computers. A couple, such as ProPilot (Sierra Corp., Bellevue, WA) and Microsoft Flight Simulator '98 provided consumers with various levels of flight training instruction. These simulators used sophisticated graphics. Realistic aircraft controls, including throttle, rudder pedals, and yoke or joystick, were easy to obtain and they added fidelity to computer simulations. Of course, the military and the airlines had numerous flight simulators, varying greatly in price and fidelity.

There were two predecessor systems chosen for analysis before designing the BFITS workstation: the T-37 aircraft (T-4) simulator and Microsoft Flight Simulator '98. During this project, the Air Force Academy owned and operated a number of T-37 simulators for procedure training. The simulators were located on the second floor Fairchild Hall, room 2C28.

The simulator itself was the cockpit of a Cessna T-37, equipped with working radios and moving instrument gauges. The simulator was run by analog computing systems and had all the moving instruments of an actual T-37. The cockpit would light up, various instruments, such as radios, had to be set, and different instruments, such as the altimeter, vertical speed indicator, and tachometer worked just as in the real aircraft. In addition, the subject in the cockpit received radio calls. In this sense, it had a very high level of fidelity.

The T-37 simulator was quite useful, for several reasons. First, it was very realistic as far as appearance. Subjects using this simulator learned where various instruments were located. Exposing a subject to this kind of instrument training may aid in memory skills and help decrease reaction time in actual flight operations. In addition, the simulated flight environment of the system was very realistic. The subjects were given a scenario to accomplish. Usually, they took off, flew various headings to navigation points, encountered air traffic, and landed the aircraft. Subjects were also given emergency scenarios to deal with, on instruments. This was an important training concept, since a person's senses do not always tell the truth in the three dimensional world of flying.

The simulator also allowed subjects to learn to listen to the radio. One of the toughest things for aviation trainees to learn is radio calls. Although they did not get to make radio calls, learning to listen for them is still an important and difficult thing to learn. Finally, the simulator did a good job of providing subjects with extremely difficult scenarios and a lot of information. Subjects could be overloaded with information. This was important because in an aircraft, emergencies can happen quickly and without much warning. The better a subject is at organizing information and reacting to difficult situations, the better he or she will react in real emergencies.

However, there were a few drawbacks to this system. One was the lack of visual feedback. The only thing the subject saw in the T-37 simulator cockpit was a white canopy. The only visual feedback was the moving instruments. Flying this simulator was similar to flying in a cloud; everything was done with instruments. Piloting by visual flight rules could not be taught. Also, there was no movement associated with the simulator. The physiological effects of pulling g's, or of even turning, could not be felt in this simulator.

Of course, when designing a flight simulator, cost is important. More realistic simulators that move cost millions of dollars. Fidelity is also important in simulator design. However, just because a simulator has more fidelity does not necessarily mean it is a better simulator. Depending on the training objective, a simulator that teaches basic procedures may well be better than a simulator that has extremely good graphics.

The other predecessor system we reviewed was Microsoft Flight Simulator '98. This was a PC based simulator. It was set up on a PC with a keyboard and joystick as control inputs. Different controls could be used with this simulator, such as a yoke and rudder pedals. This simulator also had its advantages and disadvantages. First, it also did a good job of providing subjects with realistic scenarios. Using this simulator, subjects took various academic courses that taught them various flight procedures. A faculty instructor talked to the subject while the subject was flying the simulator. The subject could opt to fly different missions with varying degrees of difficulty. For example, a subject could choose to fly into the Chicago airport with a 40-knot crosswind. By providing the subject with varying levels of emergencies, emergency procedure training could be conducted. This simulator also did a good job of providing the subject with realistic, one-way radio calls. The subject could receive the calls, but could not respond.

Unlike the T-37 simulator, this simulator had better audio and visual feedback. For audio feedback, the subject could hear various aircraft noises, such as stall warnings and different power settings. Physically, the cockpit was not as good as the T-37 simulator, but a realistic instrument panel could be seen. Also, there was good visual feedback associated with flying the aircraft. The subject could see simulated visual references on the "ground" while maintaining a level flight "picture," or see what it looked like when the aircraft was climbing, descending, or turning. This capability improved the subjects' visual flight abilities.

Just as with the T-37 simulator, however, this simulator also had its limitations. First, again, the movement of the simulator was non-existent. Besides what the subject saw on the computer screen, there was no sensation associated with moving in an aircraft. In addition, the controls were very limited. The subject could purchase more realistic controls, such as a yoke and pedals, but they were still not able to actually pull the throttle or turn a dial. The visual displays were limited. The subject did not experience the realism associated with being in an actual cockpit. Additionally, neither performance data nor training scenarios were available.

Several interesting learning issues were identified from our analyses of these two predecessor systems. The first was the importance of visual feedback from the instrument layout. Having a good instrument layout may help enhance the subject's instrument procedures. Of

course, transfer of training will be an issue when a subject leaves a simulator and steps into an actual cockpit. Second, control dynamics were important. If nothing else, at least these simulators caused the subject to consider movement in three dimensions as opposed to the typical two dimensions of most earth-based transportation, such as driving a car. Third, the simulations taught subjects to deal with information overload. Both of the predecessor systems did a good job of providing the subject with varying degrees of difficult situations. It is important teach aviation subjects to respond to the vast amount of information associated with flying, including radio calls, checklists, and trying to remember different steps to take for different situations. All of these issues may be investigated with the BFITS system.

The BFITS system was designed to teach the basic concepts and principles of flying, and then allow subjects to apply what they had learned in the simulator setting of the program. Some modifications were required to custom fit the program to the needs of personnel at the United States Air Force Academy. First, it was to be configured to support learning and training research by both USAFA cadets and faculty members. Easily understood instructions for the users were needed. In addition, pilot performance data were to be viewable in spreadsheet format. The 90-megahertz personal computer on which the program ran needed to be complemented with simulated aircraft yoke and rudder pedals that provided personal computer game joystick signals. The apparatus was to be set up in a laboratory setting (i.e., quiet environment) with ample space for the users to operate.

BFITS High-Level Task Analysis with Function Allocation

The highest task levels for BFITS operation included the following:

- Machine Setup
  - Turn on/boot up (investigator)
  - Run scandisk (investigator; DOS utility)
  - Run simulator to insure correct operation (investigator; BFITS utility)
  - Calibrate joystick yoke and rudder pedals (investigator; BFITS utility)
- Completion of informed consent process (subject and investigator)
- Lesson Accomplishment (subject)
- Recovery of Subject Data
  - Write data (BFITS automation)
  - View data (investigator; BFITS utility)
  - Convert binary flying data files to ASCII files (investigator; BFITS utility)
  - Convert lesson and flying data files to spreadsheet format (comma-delimited) files (investigator, supported by USAFA's bfit2ss.exe program)
- Move subject data to back-up medium (investigator)
- Data Reduction (investigator)
- Data Analysis and Report Writing (investigator)

Mid-Level Task Analysis

The first step in running the BFITS study is to set-up the workstation, composed of desk, adjustable chair, computer and monitor, keyboard, yoke, and foot pedals. Each of these parts of the workstation must be set up the same way each lesson for each subject. In addition,

these parts need to be set-up according to human factors principles for a simulator design (see workstation design for further detail).

The second step is to properly install the BFITS software. A c:\bfits subdirectory must be created on the hard drive. Executables, the lessons, and the flight criteria must be backed-up and copied. The third step is to test and calibrate the yoke and rudder pedals.

The next step for the investigator is to setup each subject's data disks. Each subject requires his or her own subdirectory, data files, and password. For passwords, a randomized number assignment system should be used to help assure each subject's privacy. In addition, the investigator must teach the subject how to adjust his or her chair before each sequence of academic lessons or flight simulation. The investigator must stress the importance of a consistent chair position so the subject will see and feel everything the same way during every lesson.

The investigator may, after a lesson has been accomplished, view the subject's flight summary. In this way, the investigator may verify the quality of subject's time on the system. The investigator should then view the subject's lesson summary. The BFITS software (Technology Systems, Inc., 1992) supports these functions.

The final steps for the investigator deal with converting, reducing and analyzing lesson and flying data. First, the investigator must use the BFITS software to convert the subject's flying data from binary to ASCII format. Second, the investigator should use the USAFA program, bfit2ss.exe to reduce the lesson and flying data and send it to a comma-delimited, ASCII text file that is easily imported into a spreadsheet. Third, subject data should be transferred from the hard drive to a floppy on a regular basis. More than one copy of data should be kept for each subject. After data are copied, they should be erased from the hard drive to free up storage space. Finally, the investigator should analyze the data. Finally, the investigator will complete a written report of this analysis, compiling the data into an easy-to-understand report.

Areas of Difficulty

The BFITS system contained some potential areas of difficulty related to human system operators. These were determined through observation of BFITS team members working with the software, compared to our previous studies of human factors principles. These areas of difficulty had to be considered during the preliminary design phase.

One of the first areas was the written instructions for loading the BFITS program. The instructions were not as all-inclusive as they needed to be. For example, they did not deal with the loading of CH Product controls software. For the inexperienced user, this caused undue stress and a lag between starting the program and actually being able to use the program.

Another area of difficulty was the negative transfer of training that occurred with experienced pilots. For example, the different types of aircraft they had flown caused

varying transfer effects. Fighter pilots were not used to flying with a yoke and using the rudder. However, heavy transport pilots were more comfortable with the yoke and using the rudder.

Last, extraneous noise level in the laboratory had a negative effect on the subjects. In particular, if a class was being taught, there was verbal interference while a subject was reading a BFITS lesson.

Design Resources

During our preliminary design phase (phase II), we consulted with Major Randy Gibb. Major Gibb was an experienced Air Force pilot with over 2,500 hours in the T-37, T-38, and C-5. He was also an instructor of Human Factors Engineering and Aviation Psychology, Department of Behavioral Sciences and Leadership, United States Air Force Academy. We introduced him to the BFITS and showed him what the BFITS simulation looked like. At this point in our project, we still did not have access to a stick and rudder assembly. Maj Gibb noted that the design of the simulated cockpit was acceptable for a beginner's introduction to flight training. He was a little surprised that it was a DOS program, but for this knowledge-oriented introduction to flight training there was no need for advanced graphics.

Constraints, Requirements and Possible Solutions

One major constraint of the program is the "user-friendliness" of its instructions. After booting up the computer and loading up the program, it becomes confusing how to do various tasks. For example, whenever the program is restarted, after the computer has been shut down, the flight controls need to be re-calibrated. After fumbling around for a while, we finally figured out how to correctly calibrate the controls. The procedure for calibrating the controls is not difficult. However, for the first time user, the instructions for calibration are confusing. Another example of poor user-friendliness is the process of logging off the program. To log off, the user has to go to the screen that allows a new user to begin. At this screen, the user has to type in 'byebye' under the 'user name' block, and then type in a '0' twice for the user password. After doing this, the computer returns the user to the DOS prompt. When looking through the System Reference Manual, we had to search before we could even find this procedure. This manual was disorganized and hard to understand. It was also extremely long, which caused people to not want to use it. This resulted in users who did not understanding the BFITS program and, thus, could not use the program to its full potential.

Several different solutions were possible to deal with the confusing instructions. First, automation could be used. Instead of having a paper-based reference manual, many of the instructions could be placed into the program themselves. One of the predecessor systems, Microsoft Flight Simulator '98, comes with one page of instructions; how to load the program. All the other program instructions are in the program. The program walks the user through it. However, since actually editing the BFITS program did not appear to be a possibility for this project, a paper-based instruction manual was viewed as a better option.

To create this manual, we envisioned working through the simulator and writing down instructions as they are needed, similar to a flow-chart. Instead of having to flip through the System Reference Manual, the manual we envisioned would walk the user easily through the program. This new manual would be very similar to a checklist. Once one task is accomplished, the user moves on to the next task. For example, when calibrating the controls, a checklist would show the user exactly how to calibrate.

Another constraint was the interpretation of data. After the subject had completed a lesson, the user can have the BFITS software convert the flying data to an ASCII format. However, neither the ASCII lesson nor the flying data could be used in a spreadsheet easily due to formatting limitations and the need for massive data reduction. Since the purpose of this program was to interpret data, this method needed to be much simpler. To solve this problem, we proposed creating a program to convert ASCII data automatically to a format more easily imported into a spreadsheet. This would help decrease the time needed to reduce data.

A final constraint was the time involved in going through all the simulator's lessons. There were 30 different lessons, each requiring anywhere from 15 to 30 minutes, depending upon the user's abilities. Thus, it took from 7.5 to 15 hours per subject to complete all of the lessons. This is a demanding time commitment.

To decrease the time required by the user, we would have to either decrease the number of lessons, or shorten the lessons. One way to shorten the lessons would be change some of the teaching techniques. The existing technique was to allow the user to try to answer a question twice before giving the correct answer. To shorten this, the program could provide the user with the correct answer after the first incorrect answer. However, since the purpose of the simulator was to teach, we did not view decreasing subject use time as a good option.

There were two apparent ways to deal with the investigator time demand. First, divide the lessons amongst several investigators. The other way to deal with this problem would be change the program so it would not need an investigator. If the hardware only needed to calibrated once a day, and the data needed to be collected only once a day, these might be the only times when an investigator would be needed.

We had several requirements to fulfill. First, since this was a computer based flight simulator, the computer hardware requirements were a necessity. This relatively old program required a computer with 2 megabytes (mb) of memory, 40-mb hard drive, math coprocessor, EGA/VGA color card, color monitor, joystick, game card, and rudder pedals. Another requirement relating to the hardware was the workspace for the simulator. This requirement is discussed, below. A final requirement was the time requirement. As stated already, each subject needed approximately 7.5 to 15 hours for all lessons. Not only does this take up the subject's time, but it also takes up the investigator's time. Thus, a major requirement was finding enough time when the investigator and subject can both be present.

The first two requirements, the hardware and workspace, are addressed in the Workspace Design section of this report. The third requirement, time, was addressed partially as a

constraint. To address it as a requirement, however, we simply needed to find subjects willing to spend several hours on this simulator. To make it easier on the subjects, we needed to have a system that was easy to understand, ran smoothly, and was accessible at all times. To make the system easily understood and easily operable, we need to design a good reference manual. To make it accessible at all times, the computer needed to be turned on and ready to run at any time during the day. If we were able to figure how to circumvent the requirement for an investigator at every test session, the subjects would be able to use the simulator at any practical time.

Functional Trade Study

The purpose of this functional trade discussion was to address the issue of human involvement versus automation in the BFITS system. This program did a good job of not making tasks too hard on the user. The investigator was required to turn on the computer, get it set up for the user, and interpret the data. Of course, there were more steps involved than this, but those were the basic investigator tasks. The computer, on the other hand, dealt with things such as creating data files, converting the data to ASCII, and doing all of the actual flight simulator functions. However, some functions could be changed to improve the simulator.

First, data collection should be automated. If there were a way to convert the data straight into spreadsheet form, much of the time spent by the user doing this would be saved. In addition, it would decrease the possibility of the user making errors in transferring the data, such as inadvertently deleting it. However, if this process could not be automated, a simple reference manual, which would walk the user through the process, would also be extremely beneficial.

The other task that could be automated would be command options, such as logging off the computer. With the existing system, the user did not know all the options available. For logging off, the user did not know how to do it unless they had happened to run across it in the Systems Reference Manual. This was not acceptable. The user should understand how to use the program without having to search through a manual for answers. By having the computer present the options, many those problems would be solved. However, changing the program was not practical. The other option was to create a reference manual that reads like a flow chart. For example, when the user finishes a lesson, the simple manual would show the options of going to the next lesson, or logging off.

Based upon our functional trade considerations, we viewed putting more automation into the program as the best option. However, doing this would have required us to change the actual simulator computer code. Unfortunately, no contracting funds were available for such an effort. Since this was not an option, the next best option was to create a paper-based reference manual based upon a flow chart of the simulator.

SYSTEM DESIGN

Workspace Design

The workstation was a critical aspect of the system design, providing comfort and continuity. These characteristics of the workstation design gave the subjects repeatable tactile experiences and visual cues. The essential components of the workstation included the keyboard, the yoke and pedals, the monitor, the chair, workspace (desktop), and the environment (Figure 1). We used the five male members of the BFITS team as physical models for the examinations. The average height of the members was 69.2 inches. We assumed this to be about the average of male cadets at USAFA. We reviewed all of the anthropometric design considerations using the resources specified by the State of Washington, Department of Labor and Industry (1997).

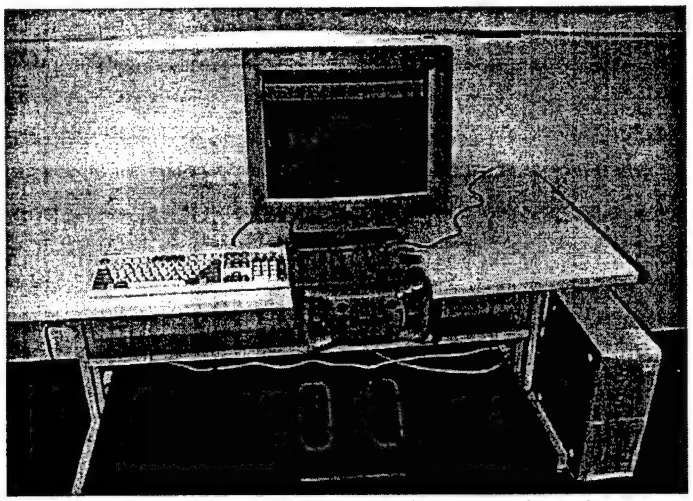

Figure 1. The BFITS simulator workstation.

The chair was adjustable so it was able to meet necessary back support requirements. The spine was supported with an extensive amount of padding in an ergonomically designed chair back. The chair height and armrest height and width were adequately adjustable. The range of vertical chair adjustability was 16.75 to 23 inches. The only problem arising from the chair was ensuring that the subjects were aware of the proper sitting position and how to attain it.

When the keyboard was in use, the wrists were in a neutral position, but there was no padding to support the wrist. This was determined not to be a problem because the lessons lasted only about twenty minutes, minimizing the likelihood of causing musculoskeletal damage. With the chair armrests in proper position, the forearms were parallel to the floor, the elbows were away from the body, and shoulders were relaxed. The keyboard was within acceptable limits based on our standards.

We analyzed the yokes and pedals using the standards for a mouse due to their functional similarities. The yoke was on the same plane as, and was acceptably close to the keyboard.

The rudder pedals were located on the floor. The operability of these items was excellent and only required minor adjustments of the springs.

The 19-inch monitor was located on a table behind the yoke. The eye declination was in the proper range of minus 15-30 degrees below horizontal gaze. The measurement for the members of our team ranged from minus 16 degrees to 25 degrees. The lateral viewing angle for all members was zero degrees. The viewing distance from the monitor to the user was an average of 33 inches. This is much greater than the recommended minimum 18 inches. The user must sit away from the table to use the yoke. In addition, the yoke attaches to the table in such a way that the screen must be pushed towards the back of the desk. At this distance, the visual image on a 15-inch screen was too small. The best way to improve this was to increase the size of the monitor.

The desktop was free of clutter and excess materials. The edges were rounded to avoid wrist strain. The table was low enough to allow access to all simulator components without resting the wrists on the table edge. The reach envelope was within anthropometric requirements. The table was still tall enough, with a vertical clearance of 27.25 inches, to allow adequate leg clearance.

The environmental analysis included lighting, temperature, and noise. The room lights were diffused to prevent reflection glare. The temperature was regulated by a central system; thus, we were unable to adjust it. However, it was comfortable for sedentary work. The sound and extra noise distractions seemed to be minimal during the design phase.

Detailed Design

There were three main design components: the computer hardware, the computer software, and paper-based instructions.

The BFITS program required at least an 80486-equivalent central processor. Due to hardware incompatibility for the newer game-card used for flight control inputs, we decided to use a Pentium 75 processor. Since this processor was faster than a 486, this computer was more than adequate for the BFITS program.

We started with a small display monitor of about 15 inches. However, after using the BFITS simulator several times and conducting a workstation analysis, we determined that we needed a bigger monitor. A 19-inch monitor proved to be big enough for acceptable BFITS operations.

We chose the CH Products game card system, including yoke and rudder pedals, due to hardware-software compatibility with the BFITS program. Also, a yoke (Model #200602, Virtual Pilot, CH Products, Poway, CA) was selected instead of a stick controller due to the applicability to the light aircraft simulation presented by the BFITS program. The software required the use of a keyboard in conjunction with the yoke for flap inputs and for the first several BFITS lessons, which required keyboard inputs. The CH products pedal assembly (Model #300-110, Pro Pedals, and ibid.) was used to control rudder inputs into the simulator.

We chose this particular rudder over a less expensive model because of its greater weight. We wanted a rudder assembly that would not move during use. We used a compatible adapter (game) card to interface the controls with the personal computer (Model 300-053, Game Card III Automatic, and ibid.).

It was determined by Koonce et al. (1995) that the breakout force of the CH Products springs in the yoke was too great. This would cause the subjects to over-compensate when making roll inputs. We replaced the yoke roll spring with one provided by Dr. Koonce. It generated less tension on the yoke. This allowed more precise minor roll adjustments of the yoke.

It was possible for us, within the BFITS software, to change the positioning of the flight instruments on the panel. However, the instruments were pre-arranged in the normal "six-pack" design that was widely accepted and used in general aviation. We showed the design to Major Gibb, introduced above. He said that the arrangement was fine for any beginner learning to fly.

A BFITS Start-Up checklist was created to help users run the program (Appendix A). The design of the checklist was similar to a pilot's checklist. It described how to log in, use the simulator, and exit the program.

We created a simple logbook since it might not be possible for an investigator to be present with the subject at all times. The logbook allowed entries of the time the subject started the program, ended the program and which lesson(s) the subject completed. The logbook allowed the investigator to monitor activities on the simulator easily, without needing access to the computer, at the DOS prompt level. Dr. Koonce (Appendix D) also provided a more complex logbook design.

To convert BFITS academic and flying data output files to spreadsheet-readable files, the program bfit2ss.exe was written by Dr. Miller in the ANSI-compatible C language (Appendix C). According to the BFITS setup, bfit2ss.exe expected the BFITS subdirectory to be in the root directory of hard drive c:, and then subject subdirectories to be within c:\bfits. The program, bfit2ss.exe, was designed to be located in c:\bfits, though other locations were acceptable. Bfit2ss is executed most simply by typing 'bfit2ss' [Enter] at the DOS prompt while in c:\bfits.

The screen output of bfit2ss showed the number of subject subdirectories detected. Then, for each subdirectory, it showed the decimal subject number, as entered by the investigator for the subject (converted from the hexadecimal code in the subdirectory name) and the subdirectory name, itself. These data were also stored in the ASCII text file, subj.txt.

Bfit2ss.exe extracted academic data from BFITS Trail files, compiling the data and preparing them to be imported into a spreadsheet. Trail files contained BFITS lesson data concerning the number of words read, the time taken for reading and response correctness. These data were stored in the comma-delimited, ASCII text file, bfit_lsn.txt. The first line of this output file showed the column headings. These data columns included the decimal subject number, the lesson number, the total time spent in the lesson, the words per minute read during the

lesson, the total number of incorrect responses for the lesson, the total number of all responses for the lesson, and, as a quality check, the number of lines of data read for the lesson from the Trail file.

The BFITS flying-data files (FCllvvv.ASC, where ll was the lesson number, and vvv was the version number [always 001, here]) contained data concerning heading, altitude, airspeed, location, etc. Bfit2ss.exe extracted flying data from these BFITS flying-data text files[1], exporting the data to similar comma-delimited ASCII text files and also reducing the data and preparing them to be imported into a spreadsheet. The data were output as similar comma-delimited FCllvvv.txt files for spreadsheet import for data validation purposes. The flying data were then also compared to criterion data (Appendix B) and reduced. The criteria were based, in turn, upon FAA requirements for light aircraft operations during the private pilot certification process.

Bfit2ss.exe stored the reduced flying data in the comma-delimited, ASCII text file, bfit_fly.txt. The first line of this output file showed the column headings. These data columns included subject number, lesson number, segment number, trial number, step number, then mean, standard deviation, number of samples, number of samples outside the respective criterion limit, and an error score for bank, heading, altitude, vertical speed, horizontal (forward) speed, altitude, and ball position, respectively.

The error score was a version of that specified by Morris and Miller (1996):

$$E = SD + ([X - A] \div W)^2$$

where E = error score,
SD = standard deviation,
X = actual mean value (for example, 22° bank)
A = assigned value (for example, 20° bank), and
W = "window" value (absolute range from assigned value; for example, ±10° bank).

For the example numbers, above, $([X - A] \div W)^2 = ([22 - 20] \div 10)^2$. However, in bfit2ss.exe, the value, $([X - A] \div W)^2$, was summed across samples. Thus, X was a sample value instead of a mean, and the equation for E was actually:

$$E = SD + \Sigma(([X - A] \div W)^2)$$

The comma-delimited data files, FCllvv.txt, bfit_lsn.txt and bfit_fly.txt, were easily imported into any spreadsheet. The bfit_lsn and bfit_fly data allowed comparisons across subject groups and across lessons.

[1] These text files were created with a BFITS Supervisor function from BFITS binary files (Technology Systems, Inc., ca. 1992).

TEST AND EVALUATION

Hypothesis

Experienced fliers will perform better than novices in basic procedures and flying skills, with a probability of 0.05 for false rejection of the null hypothesis (one-tailed). This rather obvious hypothesis allowed us to pilot test the system.

Experimental Design

The design called for two independent groups consisting of twelve subjects per group. Group one (novice) included those with no prior fixed wing flying or simulator experience. Group two (experienced) included those who had at least completed their first solo in the soaring program. Unfortunately, the actual sample sizes of the groups were quite small.

A sample power analysis of such a design follows, based upon the data from Koonce et al. (1995). In this study, the following group data were acquired with respect to student activities required to solo:

Group	Sample Size[2]	Mean no. of Student Attempted Landings	SAL Standard Deviation	Mean no. of Flight Hours Logged	Hours Standard Deviation
BFITS:	10	34.5	11.6	12.5	2.2
Control:	17	53.8	21.7	17.6	4.9
Pooled:	20 (harmonic)	46.6	18.6 (rms)	15.7	4.1 (rms)
D (sdu):		1.0		1.2	

For an expected effect size of about 1.1 sdu, and a 2-tailed t test with $\alpha = 0.05$, the power of the test would be about 92.5% (Table 2.3.5; Cohen, 1988). However, note that these measures were not available to us for analysis. We used learning measures provided by BFITS, including elapsed lesson time, words read per minute and the number of incorrect responses in quizzes. Subsequent power analyses may be conducted, based upon our report, for USAFA studies that use the learning measures.

Procedures

An investigator prepared the workspace. The table, chair, monitor, and flight controls were set-up according to human factors principles. After the workstation was set-up, the investigator turned on the computer and brought the program with a DOS batch file. The investigator also calibrated the flight controls. The program was set-up so that the subject only had to log-on and begin the lessons. In addition, a logbook was present on the desk to allow the subject to sign in and sign out.

[2] Personal communication, Dr. Koonce.

The subjects completed lessons one through nine (basic procedures) and lessons ten through fourteen (flying skills)[3]. Each lesson required approximately twenty minutes of effort by the subject. After the subjects completed a lesson, or lessons, the data were saved automatically by the BFITS software to the computer's hard drive. At the end of either the session or the day, the investigator checked the data and converted flying data to ASCII format. The subject characteristics are shown in Table 1.

TABLE 1. Subject characteristics. Class = USAFA graduation year.

Subject	Class/Rank	Gender	Age yrs	Experience	Group
1	2000	Male	21	FAA Private	Experienced
2	2000	Female	20	None	Novice
3	1999	Male	22	None	Novice
4	2000	Male	23	None	Novice
5	USAF Officer	Male		USAF > 2500 hours	Experienced
6	2000	Male	20	< 10 hours	Novice
7	USAF Officer	Male		USAF > 2500 hours	Experienced
8	1999	Male	22	6 hours	Novice
9	1999	Male	23	None	Novice
10	USAF Officer	Female		USAF pilot	Experienced
11	1999	Male	23	None	Novice

Results

Subject performance for the academic portions of BFITS is depicted in the graphs below. Because the sample size of the experienced group was so small, their data are shown as individual points related to the means and standard deviations of the novice group. The time required to complete the academics lessons, 1 through 6, did not appear to differ between novice and experienced pilots (Figure 2), nor for lessons 7 through 13 (Figure 3).

[3] The project was reviewed by USAFA Institutional Review Board (protocol no. FAC1999009) and informed consent was acquired from the subjects.

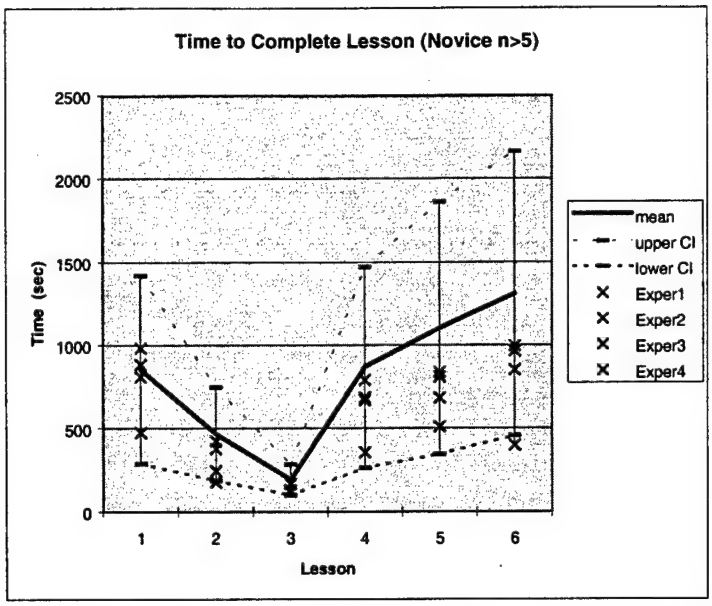

Figure 2. Time, in seconds, required to complete lessons 1 through 6. Mean (thick line) and 95% confidence limits (dotted lines) for novice pilots, with individual data points for experienced pilots (X).

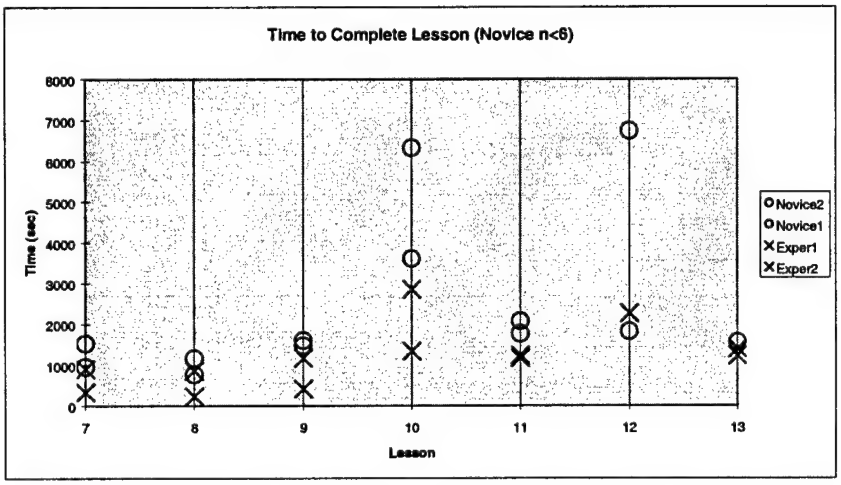

Figure 3. Time, in seconds, required to complete lessons 7 through 13. Individual data points for novice pilots (O) and individual data points for experienced pilots (X).

Experienced pilots appeared to read the academic lessons faster than novice pilots in lessons 2 and 3 (Figures 4 and 5). Novice and experienced pilots appeared not to differ on the numbers of incorrect responses to academic quizzes (Figure 6).

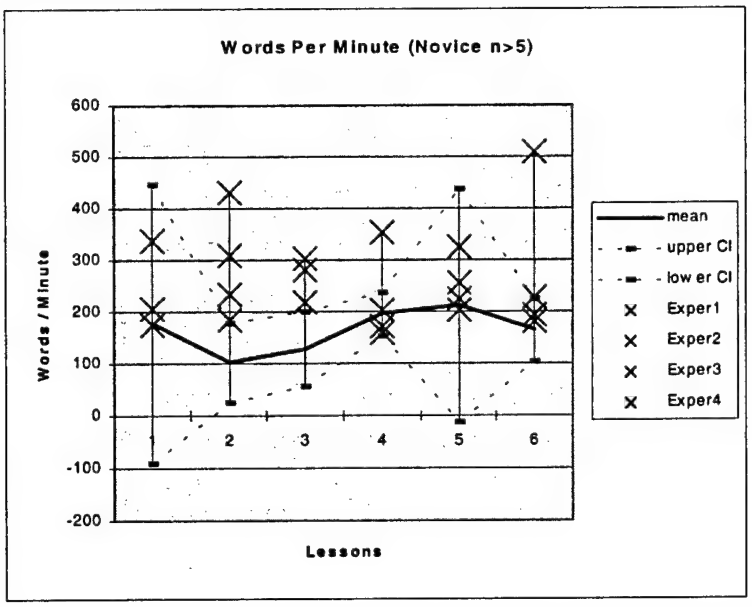

Figure 4. Reading speed, in words per minute, for lessons 1 through 6. Mean (thick line) and 95% confidence limits (dotted lines) for novice pilots, with individual data points for experienced pilots (X).

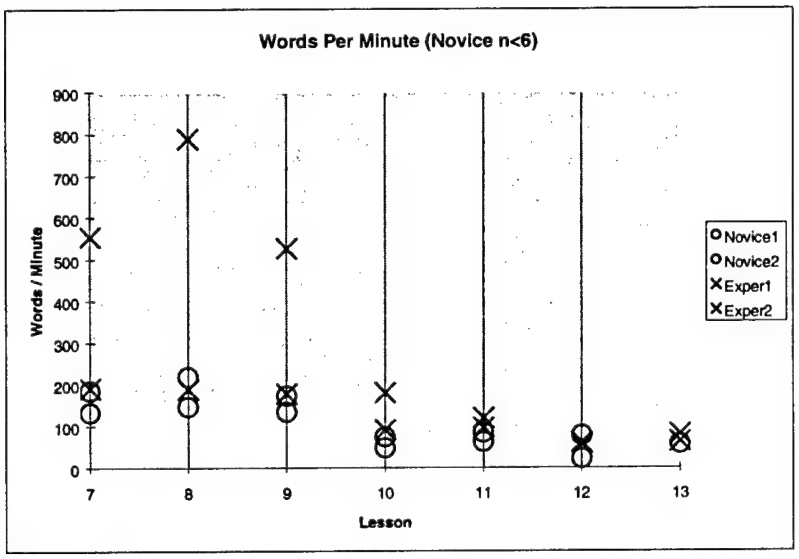

Figure 5. Reading speed, in words per minute, for lessons 7 through 13. Individual data points for novice pilots (O) and individual data points for experienced pilots (X).

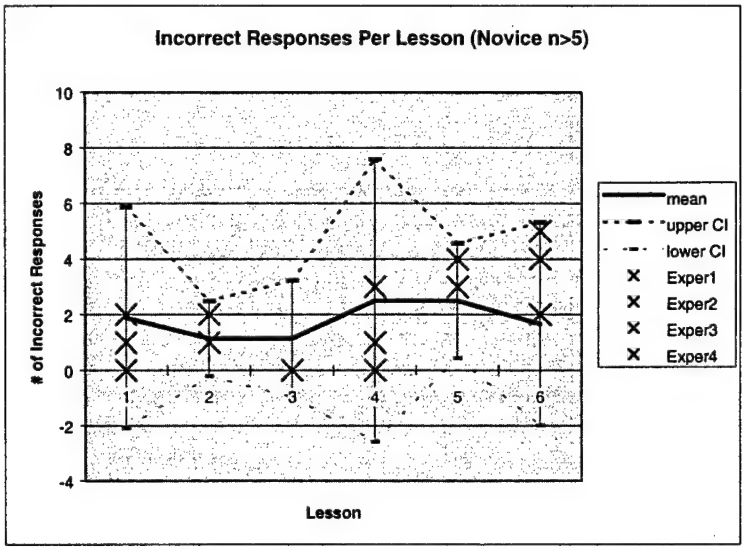

Figure 6. Numbers of incorrect responses to academic quizzes for lessons 1 through 6. Mean (thick line) and 95% confidence limits (dotted lines) for novice pilots, with individual data points for experienced pilots (X).

For the simulated flight portions of the program, we collected self-report data from the subjects. Their feedback was broken down into several different categories:

General. Experienced subjects tended to rush through the academic portions, especially the primary lessons. These subjects most likely missed questions due to carelessness or oversight that they would have answered correctly, had they taken their time. Therefore, the data (time/lesson, words/minute read, and incorrect responses) could be misleading for these subjects in lessons 1 through 9.

Aileron Control. Some experienced subjects did not use the trim function, as they preferred to simply trim for neutral and then fly the airplane more manually. Similarly, most novice subjects did not use this function. However, this was because they either were not aware of it or found it was too touchy to use.

Rudder Control. The "ball" (a simple, mechanical accelerometer that shows how well the rudders and ailerons are coordinated in a turn) was a major area of concern for experienced subjects. They felt it was too touchy, and quite unrealistic. Most of them failed several of the lessons due to being out of acceptable range in this area. Novice subjects did not have enough flying experience by which to judge the ball's feedback, but mentioned that it would "swing across" too fast in some cases, and not at all in other situations. This would induce overcorrecting, and was hard to recover. Perhaps relaxing the standards in this area is appropriate.

Throttle Control. There were not any great concerns in this area. However, some experienced subjects would have preferred a control more similar to light aircraft (pulling/pushing movement).

Flap Control. Experienced subjects would have preferred more control over the flap settings. One subject commented that in a real aircraft, one has the ability to go directly from zero degrees of flap to twenty degrees, for example.

Visual Display. This is the only form of feedback that subjects received. However, subjects felt that this was lacking in the area of realistic graphics and outside visual cues. The heading indicator raised concern among experienced subjects. They felt that an analog indicator would have been much more realistic. This would provide "moving part" feedback (Roscoe, 1968, cited in Wickens et al., 1998). Furthermore, the digital display required subjects to perform an extra cognitive step in determining their heading.

Auditory Signals. Experienced subjects felt the simulator portion of BFITS was lacking drastically in this area. The only auditory feedback they received was a rare stall warning.

Tactile Feedback. Experienced subjects felt the simulator portion of BFITS was lacking drastically in this area. Without this feedback, the simulator was much harder for these subjects to fly than their actual aircraft.

Recommendations

After completing the data analysis and gathering feedback from the subjects, we made several recommendations for BFITS. First, the quantitative portion of the test and evaluation was inconclusive. More subjects should be run to complete this effort. The other recommendations fell into three basic categories; the software, hardware, and the environment.

To begin with, fidelity was a big issue. The only feedback the user received from the simulator was visual feedback. Although the purpose of a simulator is not to re-create all aspects of feedback, there are some parts of feedback that could be added to the BFITS system. First, auditory feedback could be improved. The simulator does use some auditory feedback, but the realism of the feedback was extremely lacking. The auditory feedback should be louder and sound more like an actual airplane engine. In addition, flying cues such as a stall warning horn could be added. The simulator does sound when a stall has occurred, but a stall warning horn is supposed to sound before the stall occurs. In addition, instead of using written hints on the screen, auditory hints could be used to aid the user. Since the user is already relying heavily upon visual feedback, auditory feedback should be used. Reportedly, the latter problem has been dealt with in the follow-on version of BFITS, called the Semi-Automated Flight Evaluation System (SAFES; Baldwin et al., 1995). This version had not been made available to USAFA at the time this report was written. This was due primarily to project and personnel cutbacks in the AF Research Laboratory that led to a loss of applicable corporate memory.

The visual feedback could be improved. The current simulator did not have very good visual displays compared to many other current flight simulators. The instrument panel on the simulator also needed improvement. The heading indicator was digital instead of the typical analog display in most aircraft. In this case, an analog display is preferred because in

provides the user with more information than a digital display does. Besides the heading indicator, the "six pack" was adequate.

One final problem noted with the software was the pass-fail requirements. The biggest problem was with the rudder ball. Numerous expert subjects failed lessons due to the touchy ball. To solve this problem, the standards required to pass the lessons could be relaxed a bit. Other standards seemed to be adequate.

For the hardware, several changes also need to be made. First, the rudder controls were claimed to be touchy. To solve this problem, the springs need to be adjusted in the assembly itself. Also, there was a previously identified problem with the yoke roll spring, requiring too great a breakout force (System design, above). This would cause the user to overcompensate with control inputs. However, we substituted a spring with a lower breakout force in our BFITS workstation before we ran the test and evaluation. Finally, with current technology, force feedback joysticks are available. A force feedback joystick is simply a joystick which gives the user tactile feedback. This feedback usually occurs as the simulator takes off (the joystick shakes), when climbing in the simulator (gets stiffer as the subject climbs), and the joysticks shakes when a stall is going to occur. A force feedback yoke would be ideal for the BFITS program.

Finally, for the environment, several changes also need to be made. First, extraneous noise should be minimized. The setting we used had a notable amount of extraneous noise. To combat this problem, we provided the subjects with earmuffs. However, to really solve this problem, the earmuffs should be used in combination with a location where extraneous noise will not be encountered. Excess traffic was also a problem. People were able to walk into and out of the room while the subjects were flying. A setting should be chosen where excess traffic could either be controlled or minimized.

REFERENCES

Baldwin T, Benton C, Petriel K, Koonce JM (1995). The development of a semi-automated flight evaluation system (SAFES). *Proc 8th International Symposium on Aviation Psychology*, Ohio State Univ., Columbus, OH, pp. 1078-1081.

Benton CJ, Corriveau P, Koonce JM, Tirre WC (1992). *Development of the Basic Flight Instruction Tutoring System (BFITS).* Technology Systems, Inc., North Edgecombe, ME. Final report, contract F33615-89-C-0009, Air Force Armstrong Laboratory (AD-A246458).

Cohen J (1988). *Statistical Power Analysis for the Behavioral Sciences.* Lawrence Erlbaum, Hillsdale, NJ.

Koonce JM (1998). Use of a fully-automated criterion-referenced system for the initial training of basic pilots. *Proc. 16th biennial Applied Behavioral Sciences Symposium*, pp. 237-241, USAF Academy, CO.

Koonce JM, Moore SL, Benton CJ (1995). Initial validation of a basic flight instruction tutoring system (BFITS). *Proc 8th International Symposium on Aviation Psychology*, Ohio State Univ., Columbus, OH, pp 1037-1040.

Morris TL, Miller JC (1996). Electrooculographic and performance indices of fatigue during simulated flight. *Biological Psychology*, 42, 343-360.

Park KS, Lee SW (1992). A computer-aided aptitude test for predicting flight performance of trainees. *Human Factors*, 34, 2, 189-204.

Stark EA (1989). Simulation. Chapter 6 in R Jensen (ed.), *Aviation Psychology,* Gower Technical, Aldershot, UK.

Taylor HL, Lintern G, Koonce JM (1993). Quasi-transfer as a predictor of transfer from simulator to airplane. *J. Gen. Psychol.*, 120, 3, 257-276

Schneider W (1990). Training high performance skills: fallacies and guidelines. Chapter 21 in M Venturino, *Selected Readings in Human Factors.* Human Factors Society, Santa Monica, CA.

State of Washington (1997). *Office Ergonomics-Practical Solutions for a Safer Work Place.* Department of Labor and Industry.

Technology Systems, Inc. (ca. 1992). *System Reference Manual.* Technology Systems, Inc., Edgecombe, ME.

Wickens CD, Gordon SE, Liu Y (1998). *An Introduction to Human Factors Engineering.* Addison Wesley Longman, New York.

Appendix A

BFITS Operation Checklist for Subjects

- Complete "Name," "Date," and "Time In" portions in log notebook
- Ensure yoke/rudder pedals are connected to computer (lsn 4-14 only)
- Adjust chair for comfort and keyboard/yoke/rudder pedal access
- Turn on computer/monitor
- Type your name, press Enter
- Type last four digits of your social security number, press Enter
- Type last four digits of your social security number, press Enter (again)
- Ensure lesson number is correct
- Place earmuffs on your head
- Note: headband is adjustable
- On lesson one, read instructions on "How BFITS Works."
- Read review of last lesson completed (lsn 2-14 only)
- Read overview of current lesson
- Begin lesson, answering questions when prompted
- Note: there is scrap paper for your use in the back of the log notebook on desk
- Take quiz at end of lesson
- Note: do not worry about recording your score—this is done automatically
- Upon completion of quiz, you may either continue to next lesson or hit the Esc key to quit BFITS
- Note: unless another subject is waiting to use BFITS, turn computer off and leave monitor on
- Complete "Time Out" portion in log notebook

Appendix B

Original Criterion File for Flying Lessons

10, 1, 3, heading, 360, 10, airspeed, 80, 10, rpm, 2500, 100, ball, 0, 1, climb
10, 1, 4, heading, 360, 10, airspeed, 80, 10, ball, 0, 1, close
10, 1, 5, heading, 360, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, level off
10, 2, 2, heading, 180, 10, altitude, 3000, 100, airspeed, 80, 10, ball, cruise
10, 3, 2, open
10, 3, 3, heading, 210, 10, airspeed, 80, 10, ball, 0, 1, descent
10, 3, 4, heading, 210, 10, airspeed, 80, 10, ball, 0, 1, close
10, 3, 5, heading, 210, 10, altitude, 2000, 100, airspeed, 80, 10, rpm, 2000, 200, ball, 0, 1, level-off
11, 1, 3, bank, -15, 7, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, shallow right turn
11, 1, 4, bank, -15, 7, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, shallow right turn
11, 1, 5, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, rollout
11, 1, 6, heading, 60, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, stop turn
11, 2, 3, bank, +15, 7, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, shallow left turn
11, 2, 4, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, rollout
11, 2, 5, heading, 90, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, stop turn
11, 3, 3, bank, -30, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, medium right turn
11, 3, 4, bank, -30, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, medium right turn
11, 3, 5, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, rollout
11, 3, 6, heading, 200, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, stop turn
11, 4, 3, bank, +30, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, medium left turn
11, 4, 4, bank, +30, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, medium left turn
11, 4, 5, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, rollout
11, 4, 6, heading, 150, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, stop turn
12, 1, 3, bank, -30, 10, airspeed, 80, 10, rpm, 2500, -200, ball, 0, 1, climb right turn
12, 1, 4, airspeed, 80, 10, rpm, 2500, -200, ball, 0, 1, end turn
12, 1, 5, heading, 300, 10, airspeed, 80, 10, rpm, 2500, -200, ball, 0, 1, climbing
12, 1, 6, heading, 300, 10, airspeed, 80, 10, ball, 0, 1, level off
12, 1, 7, heading, 300, 10, altitude, 4000, 100, airspeed, 80, 10, rpm, 2000, 200, ball, 0, 1, end
12, 2, 3, bank, +30, 10, airspeed, 80, 10, ball, 0, 1, desc. left turn
12, 2, 4, bank, +30, 10, airspeed, 80, 10, ball, 0, 1, desc. left turn
12, 2, 5, airspeed, 80, 10, ball, 0, 1, end turn
12, 2, 6, heading, 300, 10, airspeed, 80, 10, ball, 0, 1, climbing
12, 2, 7, heading, 300, 10, airspeed, 80, 10, ball, 0, 1, level off
12, 2, 8, heading, 300, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, end
12, 3, 3, bank, +30, 10, airspeed, 80, 10, rpm, 2500, -200, ball, 0, 1, climb left turn
12, 3, 4, bank, +30, 10, airspeed, 80, 10, rpm, 2500, -200, ball, 0, 1, climb left turn
12, 3, 5, airspeed, 80, 10, rpm, 2500, -200, ball, 0, 1, end turn
12, 3, 6, heading, 270, 10, airspeed, 80, 10, rpm, 2500, -200, ball, 0, 1, climbing
12, 3, 7, heading, 270, 10, airspeed, 80, 10, ball, 0, 1, level off
12, 3, 8, heading, 270, 10, altitude, 4000, 100, airspeed, 80, 10, rpm, 2000, 200, ball, 0, 1, end
12, 4, 3, bank, -30, 10, airspeed, 80, 10, rpm, 1500, 200, ball, 0, 1, desc. right turn
12, 4, 4, bank, -30, 10, airspeed, 80, 10, rpm, 1500, 200, ball, 0, 1, desc. right turn
12, 4, 5, airspeed, 80, 10, rpm, 1500, 200, ball, 0, 1, end turn
12, 4, 6, heading, 90, 10, airspeed, 80, 10, rpm, 1500, 200, ball, 0, 1, descent
12, 4, 7, heading, 90, 10, airspeed, 80, 10, ball, 0, 1, level off
12, 4, 8, heading, 90, 10, altitude, 3000, 100, airspeed, 80, 10, rpm, 2000, 200, ball, 0, 1, end
13, 1, 2, start
13, 1, 3, heading, 180, 10, altitude, 3000, 100, rpm, 2500, -200, ball, 0, 1, accel. start and level
13, 1, 4, heading, 180, 10, altitude, 3000, 100, airspeed, 110, 10, ball, 0, 1, accel. start and level
13, 2, 2, start
13, 2, 3, heading, 180, 10, altitude, 3000, 100, rpm, 1750, 200, ball, 0, 1, decel. start and level
13, 2, 4, heading, 180, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, 80 mph flight

13, 3, 3, bank, +30, 10, altitude, 3000, 100, rpm 2500, -200, ball, 0, 1, accel. left turn
13, 3, 4, bank, +30, 10, altitude, 3000, 100, rpm 2500, -200, ball, 0, 1, accel. left turn
13, 3, 5, altitude, 3000, 100, ball, 0, 1, accel. left turn
13, 3, 6, bank, 0, 10, heading, 180, 10, altitude, 3000, 100, airspeed, 100, 10, ball, 0, 1, rollout
13, 4, 2, ball, 0, 1, start
13, 4, 3, bank, -30, 10, altitude, 3000, 100, rpm, 1750, 200, ball, 0, 1, decel. right turn
13, 4, 4, bank, -30, 10, altitude, 3000, 100, rpm, 1750, 200, ball, 0, 1, decel. right turn
13, 4, 5, bank, -30, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, decel. right turn
13, 4, 6, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, rollout
13, 4, 7, heading, 270, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, start and level
14, 1, 2, heading, 180, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, cruise
14, 1, 3, heading, 180, 10, altitude, 2000, 100, vertspeed, 0, 150, airspeed, 80, 10, ball, 0, 1, ?
14, 2, 3, bank, +15, 7, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, shallow left turn
14, 2, 4, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, rollout
14, 2, 5, heading, 90, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, stop turn
14, 3, 3, bank, -30, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, med. right turn
14, 3, 4, bank, -30, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, med. right turn
14, 3, 5, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, rollout
14, 3, 6, heading, 200, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, stop turn
14, 4, 3, bank, +30, 10, airspeed, 80, 10, rpm, 2500, -200, ball, 0, 1, climbing left turn
14, 4, 4, bank, +30, 10, airspeed, 80, 10, rpm, 2500, -200, ball, 0, 1, climbing left turn
14, 4, 5, airspeed, 80, 10, rpm, 2500, -200, ball, 0, 1, end turn
14, 4, 6, heading, 270, 10, airspeed, 80, 10, rpm, 2500, -200, ball, 0, 1, climbing
14, 4, 7, heading, 270, 10, airspeed, 80, 10, ball, 0, 1, level off
14, 4, 8, heading, 270, 10, altitude, 4000, 100, airspeed, 80, 10, rpm, 2000, 200, ball, 0, 1, end
14, 5, 3, bank, -30, 10, airspeed, 80, 10, rpm, 1500, 200, ball, 0, 1, desc. right turn
14, 5, 4, bank, -30, 10, airspeed, 80, 10, rpm, 1500, 200, ball, 0, 1, desc. right turn
14, 5, 5, airspeed, 80, 10, rpm, 1500, 200, ball, 0, 1, end turn
14, 5, 6, heading, 90, 10, airspeed, 80, 10, rpm, 1500, 200, ball, 0, 1, descend
14, 5, 7, heading, 90, 10, airspeed, 80, 10, ball, 0, 1, level off
14, 5, 8, heading, 90, 10, altitude, 3000, 100, airspeed, 80, 10, rpm, 2000, 200, ball, 0, 1, end
14, 6, 2, start
14, 6, 3, heading, 180, 10, altitude, 3000, 100, rpm, 2500, -200, ball, 0, 1, accel. straight and level
14, 6, 4, heading, 180, 10, altitude, 3000, 100, airspeed, 110, 10, ball, 0, 1, accel. straight and level
14, 7, 3, heading, 180, 10, altitude, 3000, 100, rpm, 1700, 200, ball, 0, 1, decel. straight and level
14, 7, 4, heading, 180, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, 80, mph flight
14, 8, 3, bank, +30, 10, altitude, 3000, 100, rpm, 2500, -200, ball, 0, 1, accel. left turn
14, 8, 4, bank, +30, 10, altitude, 3000, 100, rpm, 2500, -200, ball, 0, 1, accel. left turn
14, 8, 5, altitude, 3000, 100, ball, 0, 1, accel. left turn
14, 8, 6, bank, 0, 10, heading, 180, 10, altitude, 3000, 100, airspeed, 100, 10, ball, 0, 1, rollout
14, 9, 2, ball, 0, 1, start
14, 9, 3, bank, -30, 10, altitude, 3000, 100, rpm, 1750, 200, ball, 0, 1, decel. right turn
14, 9, 4, bank, -30, 10, altitude, 3000, 100, rpm, 1750, 200, ball, 0, 1, decel. right turn
14, 9, 5, bank, -30, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, decel. right turn
14, 9, 6, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, rollout
14, 9, 7, heading, 270, 10, altitude, 3000, 100, airspeed, 80, 10, ball, 0, 1, straight and level

Appendix C

Source Code for bfits2ss.exe (ANSI C)

```
/*
    bfit2ss.c
       Extracts data from BFITS ASCII text files for spreadsheet input.
       Created by Dr. James C. Miller, Human-Environmental Research Center,
        USAF Academy, CO 80840.  Distribution unlimited.

       22 Apr 1999:  First version; lesson data files
       16 Jun 1999:  Began addition of flying data files
       10 Feb 2000:  Began changes after data validation work
       17 Feb 2000:  Removed leading 00 in subdirectory name capture
                     Fixed the failure to close ascii flying files
                     Added dot detection in subdirectory name to exclude file names
                       from temp.txt
        18 Feb 2000:  For flying files,
                       - Corrected output lesson no. from "k" to "loop"
                       - Corrected the criterion for writing data
                       - Corrected alignment of flight criteria w/ data from k to k+1
                       - Makes comma-delimited txt file corresponding to each asc file
       3 Mar 2000:  For flying files,
                       - 999 for no ball sample (now 9999)
                       - Corrected for 2-character flap data
       22 Mar 2000: For flying files,
                       - Corrected data acquisition from asc files for blank fields; = 9999
                       - Corrected data acquisition for mismatch between steps and ET
                       - Note:  zeroes in bfit_fly.txt indicate no applicable criterion
                       - Corrected 360/0 degree heading problem in data processing
                       - Note:  criterion file north headings should be 360, not 0 degrees
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include <assert.h>
#include <direct.h>
#include <math.h>

FILE *input_file1, *input_file2, *input_file3, *output_file, *output_file2, *output_file3,
       output_file4;
char input_name1[35], input_name2[35], input_name3[35], output_name[35], output_name2[35],
       output_name3[35], output_name4[35];

char loop[3], buffer[128], subdir[128], *p;
int j, k, m, status, subs = 0, temp, lines, lesson, flying, change = 0, critlines;
long subj_no;
int total_time, total_words, total_response, total_wrong;
       /* Note:  don't use int i, M or N in main */
float wpm;
char response[2], answer[2];

struct in {                        /* For flight, criteria & failure data */
    int lessno;
    int segno;
    int trialno;
    double ET;
    double bank;
    double hdg;
    double alt;
    double vs;
    double as;
    double rpm;
    double pitch;
    double turn;
    double ball;
    double track;
    double mx;
    double mz;
```

```
    double flaps;
    int stepno;
    } read, data;

struct crit {                       /* For flight & criteria data */
    int lessno;
    int segno;
    double bank;
    double hdg;
    double alt;
    double vs;
    double as;
    double rpm;
    double ball;
    int stepno;
    } *desired, *limit;

struct _current {
        int segno;
        int trialno;
        int stepno;
        } current;

struct _error {                     /* For error scores */
    double bank_sum;
        double bank_ssq;
        double bank_n;
        double bank_out;
        double bank_E;

    double hdg_sum;
        double hdg_ssq;
        double hdg_n;
        double hdg_out;
        double hdg_E;

    double alt_sum;
        double alt_ssq;
        double alt_n;
        double alt_out;
        double alt_E;

    double vs_sum;
        double vs_ssq;
        double vs_n;
        double vs_out;
        double vs_E;

    double as_sum;
        double as_ssq;
        double as_n;
        double as_out;
        double as_E;

    double rpm_sum;
        double rpm_ssq;
        double rpm_n;
        double rpm_out;
        double rpm_E;

    double ball_sum;
        double ball_ssq;
        double ball_n;
        double ball_out;
        double ball_E;
    } error;


/* for  BASIC-like string operations */
static int stralloc_ptr;
static char *strings[8];
```

```
static int str_tag[8];

main ( int argc, char *argv[]) {

/* Title */
        clrscrn ();
    puts ( "bfit2ss.exe" );
    puts ( "Human-Environmental Research Center, 2000" );
    puts ( "USAF Academy CO 80840" );
    puts ( "Unclassified.  Distribution unlimited." );
    puts ( "*****************************************" );

/* Open flight criteria file */
        strcpy ( input_name3, "c:\\bfits\\fltcrit.txt" );
        input_file3 = fopen ( input_name3, "r");
        if ( input_file3 == NULL ) {
               puts( "*** Can't find FltCrit.txt ***");
               exit(0);
         }

/* Get criteria line count from flight criteria file */
        while ( fgets ( buffer, 128, input_file3 ) != NULL )
               critlines++;
        printf ( "%s:  number of criteria lines = %d \n", input_name3, critlines );
        rewind ( input_file3 );

/* Get flight criteria */
        desired = calloc ( critlines, sizeof ( struct crit ));
        limit = calloc ( critlines, sizeof ( struct crit ));
        for ( k = 0; k < critlines; k++ ) {
               fgets ( buffer, 128, input_file3 );
               Parse_Line ();
               }
        fclose ( input_file3 );

/* Capture subject subdirectories */
    strcpy ( buffer, "dir c:\\bfits\\* /b /o:n > c:\\bfits\\temp.txt" );
        status = system ( buffer );
        if ( status != 0 ) {
               printf ( "\noperating system error %d", status );
               exit(0);
         }
    strcpy ( input_name1, "c:\\bfits\\temp.txt" );
        input_file1 = fopen ( input_name1, "r");
        if ( input_file1 == NULL ) {
         puts( "*** Can't find temp.txt ****");
               exit(0);
         }

/* Get subject subdirectory count from temp.txt */
        while ( fgets ( buffer, 128, input_file1 ) != NULL )
               subs++;
    printf ( "%s:  number of subdirectories = %d \n", input_name1, subs );
        rewind ( input_file1 );

/* Open output files */
    strcpy ( output_name, "c:\\bfits\\bfit_lsn.txt" );
        output_file = fopen ( output_name, "w");
        fprintf ( output_file, "subject, lesson, time, wpm, incorr, responses, lines\n" );
    strcpy ( output_name2, "c:\\bfits\\bfit_fly.txt" );
    output_file2 = fopen ( output_name2, "w");
    fprintf ( output_file2, "subj, lesson, segt, trial, step, bank_mn, bank_sd, bank_n, bank_out,
        bank_E, " );
    fprintf ( output_file2, "hdg_mn, hdg_sd, hdg_n, hdg_out, hdg_E, alt_mn, alt_sd, alt_n,
        alt_out, alt_E, " );
    fprintf ( output_file2, "vs_mn, vs_sd, vs_n, vs_out, vs_E, as_mn, as_sd, as_n, as_out,
        as_E, " );
    fprintf ( output_file2, "rpm_mn, rpm_sd, rpm_n, rpm_out, rpm_E, ball_mn, ball_sd, ball_n,
        ball_out, ball_E \n" );
    strcpy ( output_name3, "c:\\bfits\\subj.txt" );
    output_file3 = fopen ( output_name3, "w");
```

```
/* Subdirectory loop */
for ( j = 0; j < subs; j++ ) {
       fgets ( buffer, 128, input_file1 );                    /* subdirectory name */
       /* exclude file name */
           strtok ( buffer, "." );
           if ( strtok ( NULL, "." )) continue;
       strcpy ( subdir, "\\bfits\\" );
       strcat ( subdir, left ( buffer, 8 ));
       subj_no = strtol ( buffer, NULL, 16 );                        /* subject number */
       if ( subj_no < 1 ) continue;
       printf ( "subject #%ld  (%s) \n", subj_no, subdir );
       fprintf ( output_file3, "\nsubject #%ld  (%s) \n", subj_no, subdir );

/* Loop for up to 31 lessons */
/* if ( subj_no > 500 ) */    /* troubleshoot */

for ( k = 0; k < 31; k++ ) {
    itoa ( k + 1, loop, 10 );
    lesson = 1; flying = 1;                        /* assume BFITS data files are present */

    /* Lesson file name and path */
    if ( k < 9 ) {
        strcpy ( buffer, "\\0" );
        strcat ( buffer, left ( loop, 1 ));
        }
    else {
        strcpy ( buffer, "\\" );
        strcat ( buffer, loop );
        }
    strcat ( buffer, "trail" );
    fnmerge ( input_name2, "c:", subdir, buffer, ".log" );
       /* printf ( "check file name for %s \n", input_name2 ); */
       input_file2 = fopen ( input_name2, "r");
    if ( input_file2 == NULL ) {
        lesson = 0;    /* no academic file */
        fclose ( input_file2 );
        }

    if ( lesson ) {
            /* printf ( "getting data from %s \n", input_name2 ); */
            fprintf ( output_file3, "getting data from %s \n", input_name2 );
            total_time = 0; total_words = 0;
            total_response = 0; total_wrong = 0; lines = 0;
        while ( fgets ( buffer, 128, input_file2 ) != NULL ) {  /* line input */
            lines += 1;
            total_words += atoi ( mid ( buffer, 9, 5 ));
            total_time += atoi ( mid ( buffer, 14, 5 ));     /* in seconds */
            strcpy ( response, mid ( buffer, 19, 1 ));
            if ( strcmp ( response, " " ) == 0 ) continue;  /* no response? */
            else {
                strcpy ( answer, mid ( buffer, 20, 1 ));
                if ( strcmp ( response, answer ) == 0 )
                    total_response += 1;                     /* correct */
                else {
                    total_wrong += 1;                        /* incorrect */
                    total_response += 1;
                    }       /* end else */
                }           /* end else */
            }               /* end of line */

        /* For all lines in lesson... */
            wpm = total_words;                                      /* float */
            wpm /= ( total_time / 60.0 );                           /* words/minute */
            fprintf ( output_file, "%4ld, %2d, %5d, %4.1f, %4d, %4d, %3d\n",
           subj_no, k + 1, total_time, wpm, total_wrong, total_response, lines );
        if ( fclose ( input_file2 ) > 0 ) printf ( "file close error" );
        }           /* end of one lesson */

    /* ASCII flying file name and path */
    if ( k > 8 ) {
```

```
    strcpy ( buffer, "\\fc" );
    strcat ( buffer, loop );
    strcat ( buffer, "001" );
    fnmerge ( input_name2, "c:", subdir, buffer, ".asc" );
    input_file2 = fopen ( input_name2, "r" );
    if ( input_file2 == NULL ) {              /* works OK 2/18/00 */
        flying = 0;    /* no flying file */
        fclose ( input_file2 );
        }
    if ( flying ) {
        fnmerge ( output_name4, "c:", subdir, buffer, ".txt" );
        output_file4 = fopen ( output_name4, "w" );

        /* printf ( "getting data from %s \n", input_name2 ); */
        fprintf ( output_file3, "getting data from %s \n", input_name2 );
        while ( fgets ( buffer, 128, input_file2 )) {              /* line input */
               Get_FlyData_Fields ();                          /* sum each line of data */
                if ( change ) {                                /* need output */
                        Step_Data_Calc ();                     /* end of step */
                        /*
                       printf ( "Subj%ld, %s, Seg%d, Tr%d, St%d \n",
                          subj_no, loop, current.segno, current.trialno, current.stepno );
                        */
                       Write_Data ();
                        current.segno = current.trialno = current.stepno = 0;
                        Init_Sums ();
                       }                                       /* end of output */

                /* look for new step info */
                if ( data.segno != 9999 ) current.segno = data.segno;
                if ( data.trialno != 9999 ) current.trialno = data.trialno;
                if ( data.stepno != 9999 ) current.stepno = data.stepno;

                /* find relevant criteria, index = m */
                for ( m = 0; m < critlines; m++ )
                if ( limit[m].lessno == ( k + 1 ) && limit[m].segno == current.segno &&
                          limit[m].stepno == current.stepno ) break;
                if ( m < critlines ) Sums ();
                }                                              /* end of line processing */
        fclose ( input_file2 );
        fclose ( output_file4 );
        }}      /* end of ASCII flying file & k > 8 */

    }           /* end of 31-lesson loop (k) */
}               /* end of subdirectory loop (j) */

        printf ( "\n\nData in %s & %s \n", output_name, output_name2 );
        fcloseall ();

}       /* end of main */


/*  **  BASIC-like string operations ** public domain by Bob Stout ** */
           /*  [ note:  1-based, like BASIC default.  JCM ] */
           /*  [ note:  uses int i, M, N ] */

static int stralloc_ptr;
static char *strings[8];
static int str_tag[8];

/*  stralloc() is the key function in this package, maintaining a pool of
    reusable strings  */
char *stralloc(int length)
{
    register int i;
    i = stralloc_ptr++;
    if ((!strings[i]) || (length > strlen(strings[i]))) {
        assert(strings[i] = (char *)realloc(strings[i], length));
        str_tag[i] = -1;
        }
```

```
    else str_tag[i] = 0;
    stralloc_ptr &= 7;
    return (strings[i]);
    /* Maintains 8 strings in a circular buffer */
}

/*  free the string pool */
void str_free(char *string)
{
    register int i;
    for (i = 0; i < 8; ++i) {
        if (strings[i] == string) {
            if (str_tag[i]) free(strings[i]);
            return;
            }
        }
}

/*  return the leftmost N characters from a string */
char *left(char *string, int N)
{
    char *buf;
    int strlength;
    strlength = strlen(string);
    if (N > strlength) N = strlength;
    buf = stralloc(N + 1);
    memcpy(buf, string, N);
    buf[N] = '\0';
    return buf;
}
/*  return the rightmost N characters from a string */
char *right(char *string, int N)
{
    char *buf;
    int strlength;
    strlength = strlen(string);
    if (N > strlength) N = strlength;
    buf = stralloc(N + 1);
    strcpy(buf, &string[strlength-N]);
    return buf;
}
/*  return a substring, N characters long beginning at position M */
char *mid(char *string, int M, int N)
{
        char *buf;
        int strlength;
        strlength = strlen(string);
        if (M > strlength) return NULL;
        if (N > (strlength - M)) N = strlength - M;
        buf = stralloc(N + 1);
        memcpy(buf, &string[M-1], N);
        buf[N] = '\0';
        return buf;
}

void Get_FlyData_Fields () {
       if ( !(strcmp ( left ( buffer, 2 ), "  " )))
              data.segno = 9999;
              else data.segno = atoi ( left ( buffer, 2 ));

       if ( !(strcmp ( mid ( buffer, 3, 2 ), "  " )))
              data.trialno = 9999;
              else data.trialno = atoi ( mid ( buffer, 3, 2 ));

       if ( !(strcmp ( mid ( buffer, 5, 6 ), "      " )))
              data.ET = 9999;
              else data.ET = atoi ( mid ( buffer, 5, 6 ));

       if ( !(strcmp ( mid ( buffer, 11, 3 ), "   " )))
              data.bank = 9999;
              else data.bank = atoi ( mid ( buffer, 11, 3 ));
```

```
        if ( !(strcmp ( mid ( buffer, 14, 3 ), "   " )))
               data.hdg = 9999;
               else data.hdg = atoi ( mid ( buffer, 14, 3 ));

        if ( !(strcmp ( mid ( buffer, 17, 5 ), "     " )))
               data.alt = 9999;
               else data.alt = atoi ( mid ( buffer, 17, 5 ));

        if ( !(strcmp ( mid ( buffer, 22, 5 ), "     " )))
               data.vs = 9999;
               else data.vs = atoi ( mid ( buffer, 22, 5 ));

        if ( !(strcmp ( mid ( buffer, 27, 3 ), "   " )))
               data.as = 9999;
               else data.as = atoi ( mid ( buffer, 27, 3 ));

        if ( !(strcmp ( mid ( buffer, 30, 4 ), "    " )))
               data.rpm = 9999;
               else data.rpm = atoi ( mid ( buffer, 30, 4 ));

        if ( !(strcmp ( mid ( buffer, 34, 3 ), "   " )))
               data.pitch = 9999;
               else data.pitch = atoi ( mid ( buffer, 34, 3 ));

        if ( !(strcmp ( mid ( buffer, 37, 3 ), "   " )))
               data.turn = 9999;
               else data.turn = atoi ( mid ( buffer, 37, 3 ));

        if ( !(strcmp ( mid ( buffer, 40, 3 ), "   " )))
               data.ball = 9999;
               else data.ball = atoi ( mid ( buffer, 40, 3 ));

        if ( !(strcmp ( mid ( buffer, 43, 1 ), " " )))
               data.track = 9999;
               else data.track = atoi ( mid ( buffer, 43, 1 ));

        if ( !(strcmp ( mid ( buffer, 44, 6 ), "      " )))
               data.mx = 9999;
               else data.mx = atoi ( mid ( buffer, 44, 6 ));

        if ( !(strcmp ( mid ( buffer, 50, 6 ), "      " )))
               data.mz = 9999;
               else data.mz = atoi ( mid ( buffer, 50, 6 ));

        if ( !(strcmp ( mid ( buffer, 56, 2 ), "  " )))
               data.flaps = 9999;
               else data.flaps = atoi ( mid ( buffer, 56, 2 ));

        if ( !(strcmp ( mid ( buffer, 58, 2 ), "  " )))
               data.stepno = 9999;
               else data.stepno =  atoi ( mid ( buffer, 58, 2 ));

    /* reset, if needed, based upon presence of new step (not ET = 000000) */
        if ( data.stepno != current.stepno && data.stepno != 9999 )
        change = 1;
        else change = 0;

    fprintf ( output_file4,
"%d,%d,%6.0f,%3.0f,%3.0f,%5.0f,%5.0f,%3.0f,%4.0f,%3.0f,%3.0f,%3.0f,%1.0f,%6.0f,%6.0f,%2.0f,%d \n",
         data.segno, data.trialno, data.ET, data.bank, data.hdg, data.alt,
         data.vs, data.as, data.rpm, data.pitch, data.turn, data.ball,
         data.track, data.mx, data.mz, data.flaps, data.stepno );
}

void Parse_Line ()
{       desired[k].lessno = limit[k].lessno = atoi ( strtok ( buffer, ", " ));
        desired[k].segno = limit[k].segno = atoi ( strtok ( NULL, ", " ));
        desired[k].stepno = limit[k].stepno = atoi ( strtok ( NULL, ", " ));
        /* printf ( "%d, %d, %d, ", desired[k].lessno, desired[k].segno, desired[k].stepno ); */
        while ( p = strtok ( NULL, ", \n" )) {
```

```
            if ( !strcmp ( p, "heading" )) {
                    desired[k].hdg = atoi ( strtok ( NULL, ", " ));
                    limit[k].hdg = atoi ( strtok ( NULL, ", " ));
                    /* printf ( "%s, %d, %d", p, desired[k].hdg, limit[k].hdg ); */
                    }
            else if ( !strcmp ( p, "vertspeed" )) {
                    desired[k].vs = atoi ( strtok ( NULL, ", " ));
                    limit[k].vs = atoi ( strtok ( NULL, ", " ));
                    /* printf ( "%s, %d, %d", p, desired[k].vs, limit[k].vs ); */
                    }
            else if ( !strcmp ( p, "airspeed" )) {
                    desired[k].as = atoi ( strtok ( NULL, ", " ));
                    limit[k].as = atoi ( strtok ( NULL, ", " ));
                    /* printf ( "%s, %d, %d", p, desired[k].as, limit[k].as ); */
                    }
            else if ( !strcmp ( p, "rpm" )) {
                    desired[k].rpm = atoi ( strtok ( NULL, ", " ));
                    limit[k].rpm = atoi ( strtok ( NULL, ", " ));
                    /* printf ( "%s, %d, %d", p, desired[k].rpm, limit[k].rpm ); */
                    }
            else if ( !strcmp ( p, "ball" )) {
                    desired[k].ball = atoi ( strtok ( NULL, ", " ));
                    limit[k].ball = atoi ( strtok ( NULL, ", " ));
                    /* printf ( "%s, %d, %d", p, desired[k].ball, limit[k].ball ); */
                    }
            else if ( !strcmp ( p, "altitude" )) {
                    desired[k].alt = atoi ( strtok ( NULL, "," ));
                    limit[k].alt = atoi ( strtok ( NULL, "," ));
                    /* printf ( "%s, %d, %d", p, desired[k].alt, limit[k].alt ); */
                    }
            else if ( !strcmp ( p, "bank" )) {
                    desired[k].bank = atoi ( strtok ( NULL, ", " ));
                    limit[k].bank = atoi ( strtok ( NULL, ", " ));
                    /* printf ( "%s, %d, %d", p, desired[k].bank, limit[k].bank ); */
                    }
            /* else printf ( "**** Undefined:<%s> *** \n", p ); */
            }
}

void Init_Sums () {
        error.bank_sum = error.bank_ssq = error.bank_E = 0.0;
        error.bank_n = error.bank_out = 0.0;

        error.hdg_sum = error.hdg_ssq = error.hdg_E = 0.0;
        error.hdg_n = error.hdg_out = 0.0;

        error.alt_sum = error.alt_ssq = error.alt_E = 0.0;
        error.alt_n = error.alt_out = 0.0;

        error.vs_sum = error.vs_ssq = error.vs_E = 0.0;
        error.vs_n = error.vs_out = 0.0;

        error.as_sum = error.as_ssq = error.as_E = 0.0;
        error.as_n = error.as_out = 0.0;

        error.rpm_sum = error.rpm_ssq = error.rpm_E = 0.0;
        error.rpm_n = error.rpm_out = error.rpm_E = 0.0;

        error.ball_sum = error.ball_ssq = error.ball_E = 0.0;
        error.ball_n = error.ball_out = 0.0;
}

void Sums () {
        float diff;

        if ( limit[m].bank && data.bank != 9999 ) {
                error.bank_sum += data.bank;
                error.bank_ssq += ( data.bank * data.bank );
                error.bank_n += 1.0;
                diff = abs ( data.bank - desired[m].bank );
                if ( diff > abs ( limit[m].bank ))  error.bank_out += 1.0;
```

```
                error.bank_E += ( diff / limit[m].bank ) * ( diff / limit[m].bank );
                }

        if ( limit[m].hdg && data.hdg != 9999 ) {
                /* heading rule for sums: use 360 degrees +/- */
                if (( desired[m].hdg >= 270 || desired[m].hdg <= 90 ) &&
                        ( data.hdg <= 90 && data.hdg >=0 ))
                        data.hdg += 360;

                error.hdg_sum += data.hdg;
                error.hdg_ssq += ( data.hdg * data.hdg );
                error.hdg_n += 1.0;
                diff = abs ( data.hdg - desired[m].hdg );
                if ( diff > abs ( limit[m].hdg ))  error.hdg_out += 1.0;
                error.hdg_E += ( diff / limit[m].hdg ) * ( diff / limit[m].hdg );
                }

        if ( limit[m].alt && data.alt != 9999 ) {
                error.alt_sum += data.alt;
                error.alt_ssq += ( data.alt * data.alt );
                error.alt_n += 1.0;
                diff = abs ( data.alt - desired[m].alt );
                if ( diff > abs ( limit[m].alt ))  error.alt_out += 1.0;
                error.alt_E += ( diff / limit[m].alt ) * ( diff / limit[m].alt );
                }

        if ( limit[m].vs && data.vs != 9999 ) {
                error.vs_sum += data.vs;
                error.vs_ssq += ( data.vs * data.vs );
                error.vs_n += 1.0;
                diff = abs ( data.vs - desired[m].vs );
                if ( diff > abs ( limit[m].vs ))  error.vs_out += 1.0;
                error.vs_E += ( diff / limit[m].vs ) * ( diff / limit[m].vs );
                }

        if ( limit[m].as && data.as != 9999 ) {
                error.as_sum += data.as;
                error.as_ssq += ( data.as * data.as );
                error.as_n += 1.0;
                diff = abs ( data.as - desired[m].as );
                if ( diff > abs ( limit[m].as ))  error.as_out += 1.0;
                error.as_E += ( diff / limit[m].as ) * ( diff / limit[m].as );
                }

        if ( limit[m].rpm && data.rpm != 9999 ) {
                error.rpm_sum += data.rpm;
                error.rpm_ssq += ( data.rpm * data.rpm );
                error.rpm_n += 1.0;
                diff = abs ( data.rpm - desired[m].rpm );
                if ( diff > abs ( limit[m].rpm ))  error.rpm_out += 1.0;
                error.rpm_E += ( diff / limit[m].rpm ) * ( diff / limit[m].rpm );
                }

        if ( limit[m].ball && data.ball != 9999 ) {
                error.ball_sum += data.ball;
                error.ball_ssq += ( data.ball * data.ball );
                error.ball_n += 1.0;
                diff = abs ( data.ball - desired[m].ball );
                if ( diff > abs ( limit[m].ball ))  error.ball_out += 1.0;
                error.ball_E += ( diff / limit[m].ball ) * ( diff / limit[m].ball );
                }
}

void Step_Data_Calc () {
        error.bank_ssq = error.bank_ssq  - ( error.bank_sum * error.bank_sum / error.bank_n );
        error.bank_ssq = sqrt ( error.bank_ssq  / error.bank_n );   /* sd */
        error.bank_sum /= error.bank_n;                                 /* mean */
        error.bank_E = error.bank_ssq + error.bank_E;                   /* error score */
        error.bank_out /= error.bank_n;                                 /* percent out */

        error.hdg_ssq = error.hdg_ssq  - ( error.hdg_sum * error.hdg_sum / error.hdg_n );
```

```
        error.hdg_ssq = sqrt ( error.hdg_ssq  / error.hdg_n );              /* sd */
        error.hdg_sum /= error.hdg_n;                                   /* mean */
        error.hdg_E = error.hdg_ssq + error.hdg_E;                      /* error score */
        error.hdg_out /= error.hdg_n;                                   /* percent out */

        error.alt_ssq = error.alt_ssq  - ( error.alt_sum * error.alt_sum / error.alt_n );
        error.alt_ssq = sqrt ( error.alt_ssq  / error.alt_n );              /* sd */
        error.alt_sum /= error.alt_n;                                   /* mean */
        error.alt_E = error.alt_ssq + error.alt_E;                      /* error score */
        error.alt_out /= error.alt_n;                                   /* percent out */

        error.vs_ssq = error.vs_ssq  - ( error.vs_sum * error.vs_sum / error.vs_n );
        error.vs_ssq = sqrt ( error.vs_ssq  / error.vs_n );             /* sd */
        error.vs_sum /= error.vs_n;                                     /* mean */
        error.vs_E = error.vs_ssq + error.vs_E;                             /* error score */
        error.vs_out /= error.vs_n;                                     /* percent out */

        error.as_ssq = error.as_ssq  - ( error.as_sum * error.as_sum / error.as_n );
        error.as_ssq = sqrt ( error.as_ssq  / error.as_n );             /* sd */
        error.as_sum /= error.as_n;                                     /* mean */
        error.as_E = error.as_ssq + error.as_E;                             /* error score */
        error.as_out /= error.as_n;                                     /* percent out */

        error.rpm_ssq = error.rpm_ssq  - ( error.rpm_sum * error.rpm_sum / error.rpm_n );
        error.rpm_ssq = sqrt ( error.rpm_ssq  / error.rpm_n );              /* sd */
        error.rpm_sum /= error.rpm_n;                                   /* mean */
        error.rpm_E = error.rpm_ssq + error.rpm_E;                      /* error score */
        error.rpm_out /= error.rpm_n;                                   /* percent out */

        error.ball_ssq = error.ball_ssq  - ( error.ball_sum * error.ball_sum / error.ball_n );
        error.ball_ssq = sqrt ( error.ball_ssq  / error.ball_n );   /* sd */
        error.ball_sum /= error.ball_n;                                     /* mean */
        error.ball_E = error.ball_ssq + error.ball_E;                       /* error score */
        error.ball_out /= error.ball_n;                                     /* percent out */
}

void Write_Data () {
        fprintf ( output_file2, "%ld, %s, %d, %d, %d, %3.3f, %3.3f, %3.0f, %3.3f, %3.3f, ",
          subj_no, loop, current.segno, current.trialno, current.stepno,
          error.bank_sum, error.bank_ssq, error.bank_n, error.bank_out, error.bank_E );

        if ( error.hdg_sum > 360 ) error.hdg_sum -= 360;

        fprintf ( output_file2, "%3.3f, %3.3f, %3.0f, %3.3f, %3.3f, %3.3f, %3.3f, %3.0f, %3.3f,
%3.3f, ",
          error.hdg_sum,  error.hdg_ssq,  error.hdg_n,  error.hdg_out,  error.hdg_E,
          error.alt_sum,  error.alt_ssq,  error.alt_n,  error.alt_out,  error.alt_E );

        fprintf ( output_file2, "%3.3f, %3.3f, %3.0f, %3.3f, %3.3f, %3.3f, %3.3f, %3.0f, %3.3f,
%3.3f, ",
          error.vs_sum,   error.vs_ssq,   error.vs_n,   error.vs_out,   error.vs_E,
          error.as_sum,   error.as_ssq,   error.as_n,   error.as_out,   error.as_E );

        fprintf ( output_file2, "%3.3f, %3.3f, %3.0f, %3.3f, %3.3f, %3.3f, %3.3f, %3.0f, %3.3f,
%3.3f \n",
          error.rpm_sum,  error.rpm_ssq,  error.rpm_n,  error.rpm_out,  error.rpm_E,
          error.ball_sum, error.ball_ssq, error.ball_n, error.ball_out, error.ball_E );
}
```

Appendix D
User Logbook

BFITS LOG
LESSONS 1-9

NAME:_____ START DATE:_____

SUBJECT CODE:_____ FINISH DATE:_____

SI=Step Information ST=Step Question C/I=Correct/incorrect TQ=Test Question

LESSON 1

	SI	SQ	C/I	TQ
1.	___	xxx	xxx	___
2.	___	___	___	___
3.	___	___	___	___
4.	___	___	___	___
5.	___	___	___	___
6.	___	___	___	___
7.	___	___	___	___
8.	___	___	___	___
9.	___	xxx	xxx	___
10.	xxx	xxx	xxx	___

LESSON 2

	SI	SQ	C/I	TQ
	___	xxx	xxx	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	xxx	xxx	___
	xxx	xxx	xxx	xxx
	xxx	xxx	xxx	xxx

LESSON 3

	SI	SQ	C/I	TQ
	___	xxx	xxx	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	xxx	xxx	___
	xxx	xxx	xxx	___
	xxx	xxx	xxx	___
	xxx	xxx	xxx	___
	xxx	xxx	xxx	___

LESSON 4

	SI	SQ	C/I	TQ
1.	___	xxx	xxx	___
2.	___	___	___	___
3.	___	___	___	___
4.	___	___	___	___
5.	___	___	___	___
6.	___	___	___	___
7.	___	___	___	___
8.	___	xxx	xxx	___
9.	xxx	xxx	xxx	___
10.	xxx	xxx	xxx	___

LESSON 5

	SI	SQ	C/I	TQ
	___	xxx	xxx	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	xxx	xxx	___
	xxx	xxx	xxx	___

LESSON 6

	SI	SQ	C/I	TQ
	___	xxx	xxxx	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	xxx	xxx	___
	xxx	xxx	xxx	___
	xxx	xxx	xxx	___

LESSON 7

	SI	SQ	C/I	TQ
1.	___	xxx	xxx	___
2.	___	___	___	___
3.	___	___	___	___
4.	___	___	___	___
5.	___	___	___	___
6.	___	___	___	___
7.	___	xxx	xxx	___
8.	xxx	xxx	xxx	___
9.	xxx	xxx	xxx	___
10.	xxx	xxx	xxx	___

LESSON 8

	SI	SQ	C/I	TQ
	___	xxx	xxx	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	xxx	xxx	___

LESSON 9

	SI	SQ	C/I	TQ
	___	xxx	xxx	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	___	___	___
	___	xxx	xxx	___
	xxx	xxx	xxx	___
	xxx	xxx	xxx	___
	xxx	xxx	xxx	___

BFITS Log

Name _____ Start Date _____

File Code # _____ Finish Date _____

Lesson	Segment	Trials
10	1	____
	2	____
	3	____
11	1	____
	2	____
	3	____
	4	____
12	1	____
	2	____
	3	____
	4	____
13	1	____
	2	____
	3	____
	4	____
14	1	y n
	2	y n
	3	y n
	4	y n
	5	y n
	6	y n
	7	y n
	8	y n
	9	y n

Lesson	Segment	Trials
16	1	____
	2	____
	3	____
	4	____
	5	____
	6	____
17	1	____
	2	____
18	1	____
	2	____
19	1	____
	2	____
20	1	y n
	2	y n
	3	y n
	4	y n
	5	y n
	6	y n
	7	y n
	8	y n
	9	y n
21	1	____
	2	____

Lesson	Segment	Trials
22	1	____
	2	____
	3	____
24	1	____
25	1	____
26	1	____
	2	____
	3	____
27	1	____
28	1	____
29	1	____
	2	____
30	1	____
31	1	y n
	2	y n
	3	y n
	4	y n
	5	y n
	6	y n
	7	y n
	8	y n
	9	y n
	10	y n
	11	y n
	12	y n